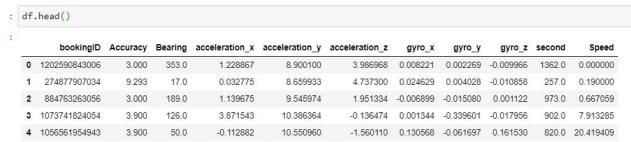
Springboard--Intermediate Data Science Program Capstone Project - Data Wrangling By Intan Sari Septiana February, 2019

## **Data Source**

The dataset was provided by Grab in this link: <a href="https://www.aiforsea.com/safety">https://www.aiforsea.com/safety</a>. It contains two folders, <a href="features">features</a> and <a href="features">labels</a>. Inside the features folder, there are 10 csv files with size 193 MB each. For the labels folder, there is one csv file with size 299 KB.

## **Data Wrangling Process**

- First, read all csv files. For features, I concat all files into one and save it as **df**. For labels. I read it as **label** variable.
- Let's take a look on each dataframe
  - o df



label

label.head()

	bookingID	label
0	111669149733	0
1	335007449205	1
2	171798691856	0
3	1520418422900	0
4	798863917116	0

- Basically they are connected by bookingID columns. df has 16135561 rows while label
  has 20018 rows, make sense because from the data description from client, one bookingID
  has thousands telematics data.
- Check the data types of each column:
  - Label

```
## check data types
label.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20018 entries, 0 to 20017
Data columns (total 2 columns):
bookingID 20018 non-null int64
label 20018 non-null int64
dtypes: int64(2)
memory usage: 312.9 KB
```

## Features

```
## check data types
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16135561 entries, 0 to 16135560
Data columns (total 11 columns):
bookingID
                 int64
                float64
Accuracy
Bearing
                float64
acceleration x float64
acceleration y float64
acceleration_z float64
                 float64
gyro x
                 float64
gyro_y
                 float64
gyro_z
                 float64
second
                 float64
Speed
dtypes: float64(10), int64(1)
memory usage: 1.3 GB
```

• Looks good and **bookingID** data type matches. Let's check null values in df, since it is clear all columns in **label** contain non null value.

```
## check null value
df.isna().sum()
bookingID
                   0
Accuracy
                   0
Bearing
acceleration x
                   0
acceleration y
acceleration_z
                   0
gyro_x
                   0
                   0
gyro_y
                   0
gyro_z
second
                   0
Speed
                   0
dtype: int64
```

• No null value. Now let's check the number of unique **bookingID** in each data frame.

```
# check unique bookingID
print('features:',df.bookingID.nunique())
print('label:',label.bookingID.nunique())
features: 20000
```

features: 20000 label: 20000

- They have the same total of unique bookingID, which does not necessarily means they
  are fully connected (we will check this later), but what we can see here, label data frame
  obviously has duplicated rows since it has <u>20018</u> rows but only has <u>20000</u> bookingID. We
  know that label only contains a pair of bookingID and its label.
- Let's try to drop the duplicated rows

```
# from above we can see that label contains 18 rows with duplicate ID, let's drop the duplicates
label = label.drop_duplicates()

label.shape
(20018, 2)
```

 After trying to drop the duplicates, the rows of label remain the same, which means they are not duplicates. We will check what is going on here but before that we will check the relationship first.

```
label bookingID in features: bookingID 20000 dtype: int64 label bookingID not in features: bookingID 0 dtype: int64
```

• From above we can say that all **bookingID** in **label** appear in **features** which means there are no **bookingID** in **label** that is not in **features**, since <u>20000</u> is the total unique **bookingID** in **label**. This should be applied to features as well. Let's confirm it.

features bookingID in label: bookingID 20000 dtype: int64 features bookingID not in label: bookingID 0 dtype: int64

It's confirmed. Let's get back to the duplicated bookingID in label. If we cannot drop it, it
means the row is not duplicated, but the data frame only has two columns, bookingID and
label. If the row is not duplicated but the bookingID is, the only possible reason is they have
different labels, which is not good. Let's check:

	bookingID	label
12602	13	1
12463	13	0
2351	154618822837	1
5295	154618822837	0
11215	223338299461	1
6212	223338299461	0
19936	395136991308	0
6121	395136991308	1
17623	403726925929	1
8472	403726925929	0
2858	455266533495	1
10778	455266533495	0

 Since the value of label with the same bookingID is contradictory, I decided to drop all of them since we do not have the true value of each bookingID. Now label only contains unique rows with unique bookingID

```
label.shape
(19982, 2)

print('label: ',label.bookingID.nunique())
label: 19982
```

• These changes should affect features too, so I decided to merge them to one data frame so deleted **bookingID** will be excluded on features and also for easier processing later.

```
print('shape: ',df.shape)
print('unique id: ',df.bookingID.nunique())
shape: (16116704, 12)
unique id: 19982
```

• This clean dataset can be used for later processing